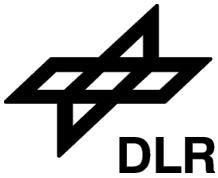

Forschungsseminar zur gleichnamigen Bachelorarbeit

Konzept und Ansatz einer Wertschöpfungskette für die Erkennung und Bereitstellung neuer Fahrumfänge intelligenter Fahrzeuge

An der
Carl von Ossietzky Universität Oldenburg
In Kooperation mit dem Deutschen Luft- und Raumfahrtzentrum
Studiengang Wirtschaftsinformatik

Vorgelegt von Linus Hestermeyer *linus.hestermeyer@gmail.com*
Matr.Nr.: 4087097

Erstprüfer: Prof. Dr. Frank Köster
Zweitprüfer: Dipl.-Inform. Gerald Sauter

Oldenburg, den December 18, 2019

Contents

Abstract

todo

Definitionen
hier die genutzten Fachwörter definieren.

1 Einführung

Mit der steigenden allgegenwärtigkeit intelligenter Fahrzeuge entwickeln sich auch die Anforderungen der Nutzer an ihr neues Auto. So entstehen neben Komfortfeatures zahlreiche Fahrtassistenzsysteme, welche zum Teil auch autonome Fahrfunktionen zu einem Fahrzeug hinzufügen. Es ist vorstellbar, in der Zukunft ist ein gestaffeltes automatischer Fahrfunktionen Seitens des Herstellers zur Verfügung zu stellen. Diese Fahrfunktionen sollen nachträglich zur Auslieferung vom Fahrer modular hinzugefügt werden können.
Halter intelligenter Fahrzeuge können dieses hierdurch zunehmend autonom fahren lassen, was den Marktwert des Autos nach dem Kauf steigern kann. Die Fahrzeughalter sollen hierbei vom System unterstützt werden in Form von Vorschlägen über neue Software, die den Bedarf des Fahrers abdeckt. Diese Vorschläge sollen zu Zeitpunkten erfolgen, in denen der Verkauf einer bestimmten Software möglichst wahrscheinlich ist. Mittels dessen wird zudem unterbunden, dass der Nutzer von zu vielen Benachrichtigungen genervt wird. Das hinzufügen von Softwarepaketen soll über eine kabellose Schnittstelle nahezu Ort- und Zeitunabhängig geschehen können um hierdurch zahlreiche Fahrten und Kosten eines Mechanikers zu ersparen und die Zugänglichkeit zu verallgegenwärtigen.

Dieses verallgegenwärtigen benötigt das Abdecken möglichst vieler Anwendungsfälle, was in diesem Fall in Form von Softwarepaketen möglich ist. Um schnellstmöglich viele Fahrfunktionen für Fahrzeughalter bereitstellen zu können, kann die Entwicklung solcher Softwarepakete durch dritte Unternehmen geschehen, welche sich explizit auf diesen Markt fokussieren. Der hierdurch entstehende Seitenmarkt für die Entwicklung von Fahrtfunktionssoftware und weiterführend möglicherweise auch Komfortfeatures kann somit schnell wachsen und sich als Teil in der Automobilindustrie etablieren. Durch autonome Fahrfunktionen werden Autos in der Regel schonender gefahren als vom Menschen, weshalb die Lebenszeit von Autos voraussichtlich verlängert wird. Ericssons Juergen Daunis sagt diesbezüglich, dass die meisten Analytiker und Führungskrfte der Meinung sind, dass der Umsatz dieser neu entstehenden Seitenmärkte weiter steigen wird und dass das traditionelle Geschäftsmodelle an dem wirtschaftlichen Maximum seiner Existenz ist. [1]

Die in dieser Arbeit erfassten Gliederungspunkte dienen als Grundlage für die gleichnamige Bachelorarbeit und sind im Wesentlichen als Literaturrecherche zu verstehen. Sie stellen wesentliche Bestandteile einzelner Bausteine der Wertschöpfungskette dar, welche in der Bachelorarbeit erarbeitet wird. Diese Wertschöpfungskette soll den Weg von der Erkennung des Bedarfs bis zu letztendlichen Softwarenutzung umfassen. Hierzu wird ein Technologiescouting durchgeführt, welches Software und Standards für unseren Anwendungsfall aufzeigt und zusätzlich auch ein Fundament für die Angebotsunterbreitung und -einholung erarbeitet. Dies geschieht in Kapitel 2 und Kapitel 3.

2 Bedarfserkennung von Software

Um Software für noch nicht zu bewältigende Verkehrssituationen zu entwickeln bedarf es der Analyse von Daten, die vom Auto während der Fahrt aufgenommen werden. Diese Daten umfassen im Fall neu zu entwickelnder Softwarepakete im wesentlichen nur eine Beschreibung der Straße. Bei der Suche nach existierenden Paketen verhält sich die ein wenig anders, weshalb im folgenden wesentliche Bausteine der Bedarfserkennung vorgestellt werden.

2.1 Bausteine der Bedarfserkennung

Um die Suche nach existierenden Paketen zu erleichtern, benötigt der Suchalgorithmus gewisse Daten als Input. Der wichtigste ist eine Beschreibung der Umwelt des Autos anhand der von den Sensoren aufgenommenen Daten *(siehe Grafik)*. Durch Fusion dieser Kamera-, Ultraschall- und Radardaten kann die Wahrnehmung der Umwelt beziehungsweise die Umgebung des Autos in einem Datenformat wie dem von der ASAM definierte Standard "OpenSCENARIO" [35] dargestellt werden, hierzu im Anschluss mehr. Ein weiterer Baustein, welcher allerdings noch nicht für intelligente Fahrzeuge zugänglich ist, ist die Car2Car-Kommunikation. Mittels dieser können Fahrzeuge untereinander kommunizieren und so in der Zukunft anderen Autos in der nähe Vorschläge für geeignete Software zu schicken. Außerdem können sich Autos in der Zukunft frühzeitig vor Gefahren warnen, was die Fahrsicherheit eines intelligenten Fahrzeugs erhöht.
Im Rahmen der Suche nach möglicherweise nötiger Software spielt die Routen- und Bewegungsplanung des Autos eine große Rolle. Zum einen kann das Auto bei der Eingabe einer neuen Route diese nach Gegenden absuchen in denen oft neue Software benötigt wird. Es kann in Folge dessen diese dem Fahrer bereits vor Fahrtbeginn vorschlagen und somit die Bequemlichkeit der Fahrt sicherstellen. Zum anderen kann das Auto auf Muster im Fahrtenverlauf achten und so bei der Erkennung eines Softwarebedarfs auf einer dieser Strecken eben diese Software zum kauf vorschlagen.

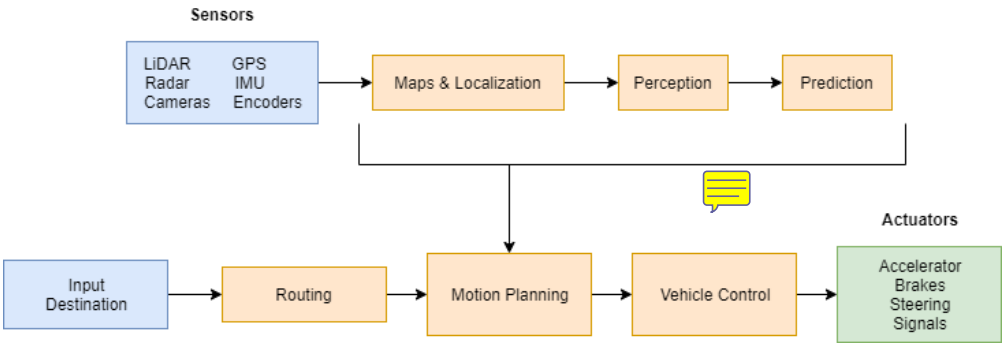

Figure 1: Architektur AV nach Jeff Schneider (CMU)

Der letzte wichtige Baustein der Bedarfserkennung sind unterstützende Suche auf einem Server. Hierdurch kann ,so lange Car2Car Kommunikation noch nicht verbreitet ist, die Region des Autofahrers und die häufigsten Strecken ähnlich wie eben beschrieben nach möglichen Softwarebedürfnissen abgesucht werden. Dieser Server und dessen Aufgaben werden im Anschluss an das folgende Kapitel beleuchtet. Vorab wird jedoch die Umweltbeschreibung mittels Sensorfusion beleuchtet, mit Fokus auf den OpenSCENARIO Standard.

2.2 Umweltbeschreibung mittels Sensorfusion

Der Inhalt dieses Kapitels basiert auf der alten Projektseite des OpenSCENARIO Standards und auf der der neuen Projektseite [35].
OpenSCENARIO arbeitet daran, dynamische von Sensoren aufgenommene Daten in ein Standard Dateiformat einzubetten. Als Basis hierzu fungieren existierende Standards zum Beschreiben statischer Umweltdaten. Durch die Fusion dieser dynamischen und statischen Daten entsteht letzten Endes ein "Szenario" in Form einer XML-basierten Beschreibung. "Der primäre Anwendungsfall von OpenSCENARIO ist es komplexe, synchrone Manöver zu beschreiben, welche mehrere Entitäten wie Fahrzeuge, Fußgänger und andere Verkehrsteilnehmer betreffen." [35]**ERSTER ABSATZ DER ASAM SEITE**

Um ein Szenario zu erstellen ist es nötig anhand von Kamera-, Ultraschall- und Radardaten alle von 11 Features von OpenSCENARIO zu evaluieren. Die Abstände zu statischen und dynamischen Objekten in der Umwelt müssen aufgezeichnet werden und die Bewegung der dynamischen Entitäten muss erfasst werden um Vorhersagen zu der Bewegung dieser tätigen zu können. Ein Szenario wird mittels eines Storyboards dargestellt, welches unterteilt werden kann in 'Stories', 'Acts' und 'Sequences'. Sequenzen können sich aus mehreren Acts und Storys zusammensetzen, während eine Story ist in mehrere acts *(Handlungen)* aufgeteilt wird. Das eigentliche Verhalten der Entitäten in den Acts wird mit Hilfe von 'events'*(d.h. wann passiert es?)* und 'actions'*(d.h. Was passiert?)* dargestellt. Wie im Anschluss erläutert werden wird, wird das OpenSCENARIO Format die Sicherheit intelligenter in Zukunft steigern. Genauer wird das anhand von Sensordaten erstellte Szenario der bisher unbekannten Situation als Input für die Suche nach passenden Softwarepaketen verwendet. Hierbei bedarf es einer Serverseitigen Auswertung dieser, um den möglichen Softwarebedarf intelligenter Fahrzeuge schnellstmöglich und in stetiger Betrachtung der Umgebung feststellen zu können.

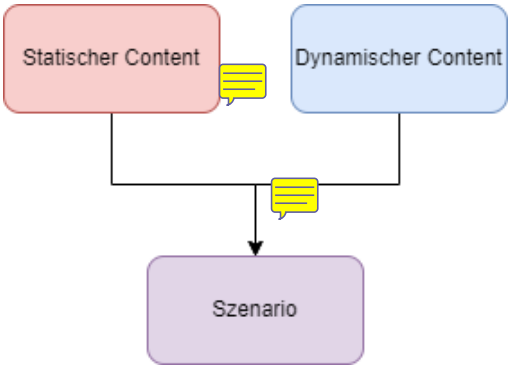

Figure 2: OpenSCENARIO Zusammensetzung

2.3 Software-User-Pattern-Recognizer [SUPR] im Rahmen der Bedarfserkennung

Das in diesem Abschnitt vorgestellte Konzept des Software-User-Pattern-Recognizer (kurz: SUPR) ist ein eigens ausgearbeiteter Baustein, welcher die Bedarfserkennung für intelligente Fahrzeuge beschleunigen und vereinfachen soll. Der Software-User-Pattern-Recognizer *(im folgenden SUPR)* soll im Rahmen der Bedarfserkennung die Suche nach passenden Softwarepaketen auf Anfrage hin durchführen **(1.)**, aber auch aktiv die Umwelt des Autos betrachten um so möglicherweise für den Fahrer relevante Softwarepakete aufzeigen**(2.)**. Beides sind Anwendungsfälle, welche auf einem Server durchgeführt werden sollten, um die Rechenleistung von Autos zu schonen.
Durch die Möglichkeit statische und dynamische Inhalte mit Hilfe des OpenSCENARIO Datenformats darstellen zu können, ist auch eine geeignete Kommunikationsgrundlage zwischen Auto und Server gegeben. Ein Auto kann ihm unbekannte Situationen beschreiben und anschließend eine Suchanfrage an den Server *(oder auch Vendor)* schicken. Auf diesem sind zwei Datenbanken vorhanden: die eine enthält sämtliche Softwarepakete für intelligente Fahrzeuge, die andere eine große Menge an OpenScenario Dateien, welche zusätzlich Fremdschlüsselverweise auf ein oder mehrere Softwarepakete bereitstellen. Das heißt dass ein unbekanntes Szenario mit Hilfe unterschiedlicher Softwarepakete bewältigt werden kann, was zusätzlich den Wettbewerb zwischen Softwareherstellern ankurbelt.
Um bei der Suche nach Software möglichst Abstrakt aber dennoch Anwendungsfall-spezifisch zu bleiben, muss eine Suche in tiefer verankerte Elemente eines Szenario gelangen, wie bspw. einzelnen acts und aber auch spezifischen Manövern. Genauer wird dieser Aspekt im folgenden beachtet und im Praxisteil der Bachelorarbeit ausgearbeitet.

1. **Gezielte Suchanfragen**

Gerät ein intelligentes Fahrzeug im Laufe seiner Fahrt in eine ihm unbekannte Situation, kann es diese mit Hilfe von OpenSCENARIO beschreiben. Das vom Auto zur unbekannten Situation erstellte Szenario wird an den Server geschickt, welcher dieses mit den auf der Datenbank liegenden Szenarien abgleicht. Diese zu entwickelnde Suche muss aufgrund der vermutlich hohen Anzahl an Daten effizient und Resourcenschonend entwickelt werden. Die Dateien müssen Muster und Gemeinsamkeiten in ihrer Struktur und im Inhalt aufweisen. Die Entwicklung eines solchen Suchalgorithmus würde den Rahmen dieser Arbeit überschreiten, weshalb im theoretischen Teil nicht mehr im Detail darauf eingegangen wird.
Ist die Suche abgeschlossen, wird eine Fallunterscheidung zwischen den folgenden Optionen getätigt:

A **Es wird kein passendes Softwarepaket gefunden**
Entweder existiert zu dem übergebenen Szenario noch keine Software, weshalb diese von Programmierern neu entwickelt werden muss. Andernfalls könnte die im Szenario dargestellte Situation noch nicht zu einer Software gemapped sein. Dies sollte von speziell hierfür angestellten Arbeitnehmern überprüft werden, um so Dopplungen in der Softwareentwicklung zu vermeiden.

B **Es wird ein passendes Softwarepaket gefunden**
Das gefundene Paket muss auf Kompatibilität mit dem Fahrzeug überprüft werden.

C **Es werden mehrere passende Softwarepakete gefunden.**
In diesem Fall gilt zu entschieden, welches Softwarepakete die besten Auswirkungen auf die Performance des Autos hat. Hierzu ist das heranziehen diverser Kennzahlen ratsam um somit die Performance der unterschiedliches Softwares untereinander zu vergleichen und eine fundierte Entscheidung treffen zu können. Eine beispielhafte

Ausarbeitung dieser Entscheidungstreffung stellt Niklas Stelter**Nikkis arbeit zitieren** in seiner Bachelorarbeit vor.

Neben der vom Fahrzeug ausgehenden Suche nach einer bestimmten Software, kann dieses auch Vorschläge für Software von einem Server erhalten, welcher dauerhaft die Umwelt von autonomen Fahrzeugen analysiert.

2. Umweltanalyse
Das Ziel hierbei ist es, die Kaufvorgänge von Softwarepaketen in der Heimatregion und entlang der gefahrenen Strecken des Autos zu überwachen. Wird z.B. entlang der geplanten Route ein häufiger Einkauf von Software XYZ festgestellt ist es möglich, dass diese auch für das eigene Auto relevant ist. Dies ist eine Software immer dann, wenn das Auto mittels dieser neue Situationen autonom bewältigen kann oder aber die Fahrsicherheit des Fahrzeugs gesteigert wird.

Figure 3: Umweltanalyse

In Grafik 3 ist folgende Situation dargestellt: Unser Auto hat als Ziel im Navigationssystem 'Braunschweig' erhalten und befindet sich aktuell auf der A2. Die roten Boxen stellen den möglichen Bereich dar, in welchem der Server die Umweltanalyse betreibt. Wird so innerhalb dieser Linien eine Software entdeckt, überprüft das System die Relevanz für das eigene Fahrzeug und schlägt sie je nach dem Vor oder nicht.

Ist die Suche *(egal ob gezielte Suche oder Umweltanaylyse)* abgeschlossen, muss die Software dem Fahrer zum Kauf vorgeschlagen und bei Bestätigung für das Auto bereitgestellt werden. Um das Fahrzeug vor Fehlern in der Software und/oder Cyber-Attacken zu schützen wird geraten, die Bereitstellung mittels Uptane(3.2) durchzuführen.

3 Bereitstellung von Software

3.1 Software Verwaltung

3.2 OTA-Updates mit UPTANE

Etwas, was nicht zu vernachlässigen ist, ist die Sicherheit der eingesetzten Software für autonome Fahrzeuge. Neu entwickelte Software hat sehr oft Bugs, wobei ein Software Bug in einem Auto tödliche Folgen haben kann. Hinzu kommt, dass Autos in der Zukunft eine große terroristische Zielscheibe sein werden. Cyberterroristen könnten Fehlfunktionen zu der Software hinzufügen, was ebenfalls tödlich enden kann. Um die Häufigkeit derartiger Aussetzer zu minimieren gilt es, eine sichere Möglichkeit für Updates zu schaffen. Neu entwickelte Software muss am besten beidseitig *(Hersteller und Auto)* verifiziert werden um so gegenüber möglichen Angriffen vorbereitet zu sein.

Erste Schritte zu einem derartigen System wurden 2010 getätigt, als Justin Samuel, Nick Mathewson, Roger Dingledine und Justin Cappos *"The Update Framework" (TUF)* entwickelten. Dieses bildet den Grundbaustein für das später entwickelte System '**Uptane**'. Mittlerweile ist es Teil des Automotive Grade Linux Projekts und somit Teil der Linux Foundation. Uptane stellt mittels einer Mehrschichtenarchitektur sicher, dass keine Schädlingssoftware auf ein Auto gelangt. Wie diese Architektur (Abb. 5) designed ist, wird in den kommenden Absätzen dargestellt.

Es ist vorab wichtig zu erwähnen, dass Uptane lediglich der Standard ist und keine offizielle Implementierung bereitstellt. Beispielhafte Implementierungen wären aktualizr, rust-tuf, Notary oder OTA Dommunity Edition. Kommerziell entwickelte Systeme werden bereitgestellt von HERE Technologies und Airbiquity.

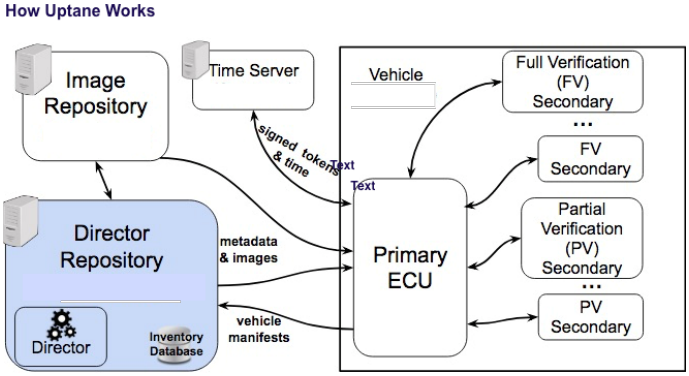

Figure 4: Architektur Uptane

Die Rechte Seite des Bildes stellen das Fahrzeug dar, während die Elemente zur Linken die Repositories verkörpern. Diese Repositories sind Server und sie haben alle eine eigene wichtige Aufgabe. Der Time-Server ist dafür da, um ECUs über die aktuelle Zeit zu informieren, da viele ECUs keine Uhr haben. Das Image Repository speichert jedes derzeit vom OEM verteilte Image zusammen mit den Meta-Daten, welche zur Authentizität benötigt werden. Es nutzt offline Schlüssel um alle metadaten zu 'unterschreiben' bzw. zu verifizieren, was einen hohen Sicherheitsvorteil darstellt. Das Director Repository entscheidet abhängig von den übergebenen Informationen des Autos genau, welche Images an die ECUs verteilt werden müssen. Es nutzt Online Schlüssel zum unterschreiben der Metadaten. Hierdurch haben OEMs die Möglichkeit, schnell neue Software auf den Markt zu bringen. Ein Auto hat mehrere ECUs, welche sich in Speicherplatzgröße, Stromverbrauch und Aufgabenbereich unterscheiden.

In dem ersten Schritt eines Updates schickt das Fahrzeug sein Manifest an das Director repository. Dieses enthält informationen darüber, welche Images aktuall auf dem Auto installiert sind. Das Director Repository entscheidet anhand dessen, welche Software auf dem Fahrzeug installiert(geupdated) werden soll. Die Metadaten und die neuen Images werden zurück an die ECU geschickt. Hier findet zunächst eine Verifikation der neuen Images statt, bevor sie anschließend bei Erfolgreichem Test installiert werden. Die Verifikation der ECUs kann entweder vollständig oder teilweise erfolgen. **Vollständig** heißt in diesem Kontext, dass die Größe der Software und die Hashes, welche die ECU über die Metadaten vom Director Repository erhält, identisch sind zu denen der Metadaten die vom Image Repository zur Verfügung gestellt werden. Bei der **teilweisen** Verifikation muss lediglich die Signatur der Metadaten des Director Repositories mit der Signatur der Metadaten vom Image Repository übereinstimmen.

einarbeiten:
stetig Updates für bereits verteilte Software nachladen

3.3 SUPR im Rahmen der Softwarebereitstellung

Der Software-User-Pattern-Recognizer *(im folgenden SUPR)* soll im Rahmen der Bereitstellung...

- SW-Vorschläge angebracht vorschlagen... [Server]
- Bestimmen der geeigneten Verkaufsweise [Server]
- Preisbestimmung je nach Nutzer und Kaufart [Server]
- Stresslevel einschätzung [Auto]
- Kaufbereitschaft einschätzen [Auto]
- Unterscheidung zwischen einzelnen fahrern des gleichen Autos, bzgl. dessen welche SW bereitgestellt wird
- möglich nötige SW für Fahrzeug soll bereits vor Kauf als Blaupause oder sogar komprimiert heruntergeladen sein, um schnellstmöglich installiert werdne zu können. Dies können auch mehrere SW-Pakete sein.

3.4 Angebotsunterbreitung & -einholung

3.4.1 Psychologische Aspekte der Verkaufspolitik (?)

4 Technologiescouting und Methodik

UCD
Rcihtlinien HCD
Carla
IntelliJ, Eclipse, AndroidStudio
Android Framework

References

[1] Daunis, Juergen: 'The automotive industry in transformation – business model disruption'
https://www.ericsson.com/en/blog/2017/11/the-automotive-industry-in-transformation–business-model-disruption (abgerufen am 14.11.2019).

[2] Logistik-info.de: 'Push und Pull Prinzip der Logistik'
https://www.logistik-info.net/diverses/push-vs-pull-prinzip/ (abgerufen am 16.11.2019)

[3] Volkswagen: 'Modelle'
https://www.volkswagen.de/de/modelle-und-konfigurator.html (abgerufen am 16.11.2019)

[4] Personas QUelle definition

[5] Lex Fridman, MIT: 'Human-Centered Autonomous Vehicle Systems: Principles of Effective Shared Autonomy'

[6] Lex Fridman: https://lexfridman.com/ (abgerufen am 17.11.2019)

[7] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. 2009. e DARPA urban challenge: autonomous vehicles in city traffic. Vol. 56. springer.

[8] Lex Fridman, Daniel E. Brown, Michael Glazer, William Angell, Spencer Dodd, Benedikt Jenik, Jack Terwilliger, Julia Kindelsberger, Li Ding, Sean Seaman, Hillary Abraham, Alea Mehler, Andrew Sipperley, Anthony Pettinato, Linda Angell, Bruce Mehler, and Bryan Reimer. 2017. MIT Autonomous Vehicle Technology Study: LargeScale Deep Learning Based Analysis of Driver Behavior and Interaction with Automation. CoRR abs/1

[9] Kieren McCarthy. 2018. So when can you get in the first self-driving car? GM says 2019. Mobileye says 2021. Waymo says 2018. (May 2018). https://www.theregister.co.uk/2018/05/09/first autonomous vehicles/

[10] Alex Davies. 2015. Oh Look, More Evidence Humans Shouldn't Be Driving. (May 2015). https://www.wired.com/2015/05/ oh-look-evidence-humans-shouldnt-driving/ [9] Terrence Fong, Charles Thorpe, and Charles Baur. 2

[11] Tom Vanderbilt and Brian Brenner. 2009. Traffic: Why We Drive the Way We Do (and What It Says about Us) , Alfred A. Knopf, New York, 2008; 978-0-307-26478-7. (2009).

[12] Raja Parasuraman and Victor Riley. 1997. Humans and automation: Use, misuse, disuse, abuse. Human factors 39, 2 (1997), 230–253.

[13] Thomas B Sheridan. 2002. Humans and automation: System design and research issues. Human Factors and Ergonomics Society.

[14] ADAC: https://www.adac.de/rund-ums-fahrzeug/ausstattung-technik-zubehoer/autonomes-fahren/grundlagen/autonomes-fahren-5-stufen/ (aufgerufen 17.11.2019)

[15] Markus I. Reinke (2013): '30 Minuten Verkaufspsychologie'

[16] S. Debernard et. al.2016: 'Designing Human-Machine Interface for Autonomous Vehicles'

[17] https://medium.com/divya-krishnan-design/ui-ux-for-autonomous-vehicle-interface-to-build-trust-de7f4c545c3b

[18] https://medium.com/punchcut/ux-design-for-autonomous-vehicles-9624c5a0a28f

[19] https://link.springer.com/content/pdf/10.1007%2F978-0-85729-085-4.pdf, Seiten 1271-1310

[20] https://iopscience.iop.org/article/10.1088/1757-899X/252/1/012096/pdf

[21] Bertoncello M and Wee D 2015 Ten ways autonomous driving could redefine the automotive world. McKinsey&Company http://www.mckinsey.com/industries/automotive-andassembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world

[22] https://www.gruenderszene.de/lexikon/begriffe/usability?interstitial

[23] https://www.usability.gov/what-and-why/user-centered-design.html

[24] https://www.seobility.net/de/wiki/User_Centered_Designs

[25] Automatisierung - Von Fahrerassistenzsystemen zum automatisierten Fahren - VDA

[26] https://www.focus.de/auto/autoentwicklung/bellbergs-auto-klartext-sorgen-autonome-autos-fuer-das-bedingungslose-grundeinkommen_id_10794360.html

[27] https://www.manager-magazin.de/unternehmen/autoindustrie/selbstfahrendes-kfz-mensch-vs-maschine-wer-gefahren-besser-erkennt-a-1200362.html

[28] https://www.autonomes-fahren.de/musk-tesla-faehrt-autonom-zum-ende-des-jahres/

[29] https://waymo.com/lidar/

[30] Christian Ress, Martin Wiecker - Potenzial der V2X-Kommunikation für Verkehrssicherheit und Effizienz; (im abstract)

[31] Brian Reimer, PhD; MIT (ZITAT Youtube Video)

[32] https://t3n.de/news/tesla-statistik-unfaelle-autopilot-1136864/

[33] UPTANE - Security and Customizability of Software Updates for Vehicles; Trishank Karthik Kuppusamy, Lois Anne DeLong and Justin Cappos

[34] https://www.youtube.com/watch?v=jTio_MPQRYc&t, Jeff Schneider Vortrag

[35] https://www.asam.net/standards/detail/openscenario/ http://www.openscenario.org/docs/IBSWorkshop20150609_VIRES_public.pdf